***1.using user define function check age is valid or not for voting***

class InvalidAgeException extends Exception

{

public String toString()

{

return "Sorry, Age must be greater than 18";

}

}

class Main

{

public static void main(String[] args)

{

int age=15;

try

{

if (age < 18)

{

throw new InvalidAgeException();

}

else

{

System.out.println("Eligible for voting");

}

}

catch (InvalidAgeException e1)

{

System.out.println(e1);

}

System.out.println("Rest of the code");

}

}

***Output:***

Sorry, Age must be greater than 18

Rest of the code

***2. Write a java program to accept a number from the user, if number is less than zero then throw user defined Exception “Number is 0” otherwise calculate the sum of first and last digit of a given number.***

import java.util.\*;

class ZeroException extends Exception

{ public String toString()

{

return "no is less than or equal to 0";

}

}

public class Main

{

public static void main(String[] args)

{

int n,first=0,last,sum=0;

Scanner sc=new Scanner(System.in);

System.out.println("Enter 1 no");

n=sc.nextInt();

try

{

if(n<=0)

{

throw new ZeroException ();

}

else

{

last=n%10;

while(n>0)

{

first=n;

n=n/10;

}

sum=first+last;

System.out.println("Sum="+sum);

}

}

catch(ZeroException e)

{

System.out.println(e);

}

}

}

Output:

Enter 1 no

123

Sum=4

3. **Accept name from user character whether is valid or not using Exception handling.**

import java.util.\*;

class NameException extends Exception

{

public String toString()

{

return "name is invalid";

}

}

public class Main

{

public static void main(String[] args)

{

int cnt=0;

Scanner sc=new Scanner(System.in);

System.out.println("Enter name");

String s1=sc.next();

try

{

for(int i=0;i<s1.length();i++)

{

char ch=s1.charAt(i);

if((ch>='a' && ch<='z')||(ch>='A' && ch<='Z'))

{

cnt++;

}

else

{

throw new NameException();

}

}

}

catch(NameException e)

{

System.out.println(e);

}

if(cnt==s1.length())

{

System.out.println("name is valid");

}

}

}

***Output:***

Enter name

radha

name is valid

***4. Create a class Student with attributes rollno, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception “AgeNotWithinRangeException”. If name contains numbers or special symbols raise exception “NameNotValidException”. Define the two exception classes.***

import java.util.\*;

class AgeNotWithinRangeException extends Exception

{

public String toString()

{

return "Age is not in range";

}

}

class NameNotValidException extends Exception

{

public String toString()

{

return "name is invalid";

}

}

class CourseNotValidException extends Exception

{

public String toString()

{

return "course is invalid";

}

}

class Student

{

int rollno,age; String name,course;

Student(int rollno,int age, String name,String course)

{

this.rollno=rollno;

this.age=age;

this.name=name;

this.course=course;

}

void display()

{

System.out.println("roll no:"+rollno);

System.out.println("age:"+age);

System.out.println("name:"+name);

System.out.println("course:"+course);

}

}

class Main

{

public static void main (String[] args)

{

int rollno,age,cnt=0;

String name,course;

Scanner sc=new Scanner(System.in);

System.out.println("enter the rollno,age,name,course");

rollno=sc.nextInt();

age=sc.nextInt();

name=sc.next();

course=sc.next();

try

{

if(age>=15 && age<=21)

{

for(int i=0;i<name.length();i++)

{

char ch=name.charAt(i);

if(Character.isLetter(ch))

{

cnt++;

}

else

{

throw new NameNotValidException ();

}

}

if(cnt==name.length())

{

Student s1=new Student(rollno,age,name,course);

s1.display();

}

}

else

{

throw new AgeNotWithinRangeException();

}

}

catch(AgeNotWithinRangeException e)

{

System.out.println(e);

}

catch(NameNotValidException e1)

{

System.out.println(e1);

}

//catch(CourseNotValidException e2)

//{

// System.out.println(e2);

//}

}

}

***Output:***

enter the rollno,age,name,course

12

23

nitya

bca

Age is not in range

***5. Define Exceptions VowelException ,BlankException, ExitException. Write another class Test which reads a character. If it is a vowel, throw VowelException,if it is blank throw BlankException and for a character 'X' throw an ExitException and terminate program. For any other character, display “Valid character”.***

import java.util.Scanner;

class VowelException extends Exception {

public VowelException() {

super("Vowel encountered");

}

}

class BlankException extends Exception {

public BlankException() {

super("Blank character encountered");

}

}

class ExitException extends Exception {

public ExitException() {

super("Exit requested");

}

}

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

char ch;

try {

System.out.print("Enter a character: ");

ch = scanner.next().charAt(0);

if (Character.isWhitespace(ch)) {

throw new BlankException();

} else if (ch == 'X' || ch == 'x') {

throw new ExitException();

} else if (Character.toLowerCase(ch) == 'a' ||

Character.toLowerCase(ch) == 'e' ||

Character.toLowerCase(ch) == 'i' ||

Character.toLowerCase(ch) == 'o' ||

Character.toLowerCase(ch) == 'u') {

throw new VowelException();

} else {

System.out.println("Valid character");

}

} catch (VowelException e) {

System.out.println(e.getMessage());

} catch (BlankException e) {

System.out.println(e.getMessage());

} catch (ExitException e) {

System.out.println(e.getMessage());

System.exit(0); // Terminate the program

}

}

}

***Output:***

Enter a character: a

Vowel encountered

***6. Define class MyDate with members day, month, year. Define default and parameterized constructors. create a date object. Throw user defined exceptions – “InvalidDayException” or “InvalidMonthException” if the day and month are invalid. If the date is valid, display message “Valid date”***

import java.util.\*;

class MyDate

{

int day,month,year;

MyDate()

{

System.out.println("\nNo Date Initialised...");

}

MyDate(int d,int m,int y)

{

day=d;

month=m;

year=y;

}

boolean checkmonth()

{

if(month<1 || month >12)

return false;

else

return true;

}

boolean checkday()

{

boolean state;

if(day<1 || day>31 )

state=false;

else

{

switch (month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

if(day > 31)

{

state= false;

}

else

state= true;break;

case 4:

case 6:

case 9:

case 11:

if(day > 30)

{

state= false;

}

else

state= true;break;

case 2:

if(year % 4 != 0 && day > 28)

{

state= false;

}

else

return true;

if(day > 29)

{

state= false;

}

default:

state= false;

}

}

return state;

}

}

class InvalidDayException extends Exception

{

public InvalidDayException(String s)

{

super(s);

}

}

class InvalidMonthException extends Exception

{

public InvalidMonthException(String s)

{

super(s);

}

}

class Main

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int d1,m1,y1;

System.out.println("enter day,year,month:");

d1=sc.nextInt();

y1=sc.nextInt();

m1=sc.nextInt();

MyDate a =new MyDate(d1,m1,y1);

try

{

if(! a.checkmonth())

throw new InvalidMonthException("Month is Invalid");

if(! a.checkday())

throw new InvalidDayException("Day is Invalid");

System.out.println("\n Valid Date....");

}

catch(InvalidMonthException e)

{

System.out.println(e);

}

catch(InvalidDayException e)

{

System.out.println(e);

}

}

}

***Output:***

enter day,year,month:

3

2023

6

Valid Date....

***7. Write a program which accept two integers and an arithmetic operator from the user and performs the operation. Fire the following user defined exceptions:***

***i. If the no of arguments are less than 3 then fire “IllegalNumberOfArguments”***

***ii. If the operator is not an Arithmetic operator, throw “InvalidOperatorException”.***

***iii. If result is –ve, then throw “NegativeResultException”***

import java.util.Scanner;// Exception for illegal number of arguments

class IllegalNumberOfArguments extends Exception {

public IllegalNumberOfArguments(String message) {

super(message);

}

}

// Exception for invalid arithmetic operator

class InvalidOperatorException extends Exception {

public InvalidOperatorException(String message) {

super(message);

}

}

// Exception for negative result

class NegativeResultException extends Exception {

public NegativeResultException(String message) {

super(message);

}

}

public class Main {

public static void main(String[] args) {

if (args.length < 3) {

try {

throw new IllegalNumberOfArguments("Not enough arguments provided. Please provide two integers and an operator.");

} catch (IllegalNumberOfArguments e) {

System.out.println(e.getMessage());

return;

}

}

try {

int num1 = Integer.parseInt(args[0]);

int num2 = Integer.parseInt(args[1]);

String operator = args[2];

int result;

switch (operator) {

case "+":

result = num1 + num2;

break;

case "-":

result = num1 - num2;

break;

case "\*":

result = num1 \* num2;

break;

case "/":

if (num2 == 0) {

throw new ArithmeticException("Division by zero is not allowed.");

}

result = num1 / num2;

break;

default:

throw new InvalidOperatorException("Invalid operator. Please use +, -, \*, or /.");

}

if (result < 0) {

throw new NegativeResultException("Result is negative: " + result);

}

System.out.println("Result: " + result);

} catch (NumberFormatException e) {

System.out.println("Invalid number format. Please enter valid integers.");

} catch (InvalidOperatorException | NegativeResultException e) {

System.out.println(e.getMessage());

}

}

}

***Output:***

Not enough arguments provided. Please provide two integers and an operator.

***8. Define a class which contains method “DisplayColor” which takes one character as argument. Raise an error if the character is not an alphabet. If the alphabet is a color of the rainbow, display the color name. If it is any other alphabet, report an error.***

public class Main {

public void displayColor(char color) {

String colors = "ROYGBIV";

if (!Character.isLetter(color)) {

throw new IllegalArgumentException("Input must be an alphabet");

}

color = Character.toUpperCase(color);

if (colors.indexOf(color) != -1) {

switch (color) {

case 'R':

System.out.println("Red");

break;

case 'O':

System.out.println("Orange");

break;

case 'Y':

System.out.println("Yellow");

break;

case 'G':

System.out.println("Green");

break;

case 'B':

System.out.println("Blue");

break;

case 'I':

System.out.println("Indigo");

break;

case 'V':

System.out.println("Violet");

break;

}

} else {

throw new IllegalArgumentException("Input alphabet is not a rainbow color");

}

}

public static void main(String[] args) {

Main rainbowColor = new Main();

try {

rainbowColor.displayColor('R');

rainbowColor.displayColor('Z');

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

}

}

***Output:***

Red

Input alphabet is not a rainbow color

***9. Define class EmailId with members ,username and password. Define default and parameterized constructors. Accept values from user throw defined exceptions – “InvalidUsernameException” or “InvalidPasswordException” if the username and password are invalid.***

***If(uname.equals(“”)||!(uname.equals(“admin”)))***

import java.util.Scanner;

class InvalidUsernameException extends Exception {

public InvalidUsernameException(String message) {

super(message);

}

}

class InvalidPasswordException extends Exception {

public InvalidPasswordException(String message) {

super(message);

}

}

class EmailId {

String username;

String password;

public EmailId() {

this.username = "";

this.password = "";

}

public EmailId(String username, String password) throws InvalidUsernameException, InvalidPasswordException {

if (username.equals("") || !username.equals("admin")) {

throw new InvalidUsernameException("Invalid username");

}

if (password.length() < 8) {

throw new InvalidPasswordException("Password must be at least 8 characters");

}

this.username = username;

this.password = password;

}

}

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter username: ");

String username = scanner.next();

System.out.print("Enter password: ");

String password = scanner.next();

EmailId emailId = new EmailId(username, password);

System.out.println("EmailId created successfully");

} catch (InvalidUsernameException e) {

System.out.println(e.getMessage());

} catch (InvalidPasswordException e) {

System.out.println(e.getMessage());

}

}

}

***Output:***

Enter username: admin

Enter password: 123456789

EmailId created successfully

***10. Define a class patient (patient\_name, patient\_age, patient\_oxy\_level,patient\_HRCT\_report). Create an object of patient. Handle appropriate exception while patient oxygen level less than 95% and HRCT scan report greater than 10, then throw user defined Exception “Patient is Covid Positive(+) and Need to Hospitalized” otherwise display its information.***

class CovidPositiveException extends Exception {

public CovidPositiveException(String message) {

super(message);

}

}

class Patient {

String patientName;

int patientAge;

double patientOxyLevel;

int patientHRCTReport;

public Patient(String patientName, int patientAge, double patientOxyLevel, int patientHRCTReport) throws CovidPositiveException {

this.patientName = patientName;

this.patientAge = patientAge;

this.patientOxyLevel = patientOxyLevel;

this.patientHRCTReport = patientHRCTReport;

if (patientOxyLevel < 95 && patientHRCTReport > 10) {

throw new CovidPositiveException("Patient is Covid Positive(+) and Need to be Hospitalized");

}

}

public void displayInfo() {

System.out.println("Patient Name: " + patientName);

System.out.println("Patient Age: " + patientAge);

System.out.println("Patient Oxygen Level: " + patientOxyLevel);

System.out.println("Patient HRCT Report: " + patientHRCTReport);

}

}

class Main{

public static void main(String[] args) {

try {

Patient patient = new Patient("John Doe", 30, 94, 12);

// Display patient information

patient.displayInfo();

} catch (CovidPositiveException e) {

System.out.println(e.getMessage());

}

}

}

***Output:***

Patient is Covid Positive(+) and Need to be Hospitalized